Chapter 10

Programming in C

Lesson 03

Array, Structure, Union and Pointer

Array

- One-dimensional array
- Multi-dimensional arrays of the primitive data types

unsigned character array for 20 bytes received at a port

- The array declaration —
  unsigned character data PortBytes [20]; The PortBytes is an array with 20 members with memory type data (direct addressable internal data memory)

unsigned character array for 20 bytes received at a port

- Each member saved as two-byte integer number
- The unsigned character variable requires 16-bit memory in SDCC compiler

unsigned character array for 20 bytes received at a port

- The size of array in memory = 20 × 1 B = 20 B.
- The array has a base address. Each member is at the address = (base address + i], assume that i is an unsigned integer and i can take the values = 0, 1, 2, 3, ……, or 19.

unsigned character array for 20 bytes received at a port

- The PortBytes [0] refers to first byte, The PortBytes [1] refers to second byte. The PortBytes [i] refers to (i + 1)th byte.

unsigned character array for 20 bytes received at a port

- The PortBytes [2× i] refers to (2× i + 1) th byte
- Expression permitted within square bracket provided the result of expression defines the 1st or 2nd or ….. 20th member.
- The PortBytes [19] refers to last member the 20th byte.

String as a Variable Size Array of Character

char code projectName[] = "Serial Port Program:"; /* projectName is an array, it is of code memory type and the variable data-type is of character.

/*The projectName has a base address, which points to the string. */

/*The memory required by an string = number of characters plus 1 Byte. The last byte of a string corresponds to NULL character.*/

String as a Variable Size Array of Character

- unsigned long xdata timeArray[10]; /* timeArray is an array of length 10, it is of xdata memory type, and the variable data-type is of unsigned long (32-bit)

The memory required = 40 B

This long integer is of 32-bit in SDCC compiler. */

String as a Variable Size Array of Character

- unsigned char xdata LCDData [8][4][2]; /* LCDData is a three dimensional array 8×4×4 variable, it is of xdata memory type, and the data-type is of unsigned char (8-bit) */

Struct

```
• struct {
      char xdata projectName[] : "Serial Port
  Program:";
      char  : 2;
      int data rate : 9600;
    } new = {“LED Blinking Program”, 4};
```

Struct

/* new.projectName = {“LED Blinking Program”, new.rate = 4

Version SDCC 2.9.4

- The values in a structure may be left un-initialised in ANSI C99 standard
- For example, character is not assigned any variable name
- Hence the field for char is empty in the structure *new*
- In old versions— not permissible
- The rate —will remain un-initialised in old versions

Union

- Specified in a way similar to a struct
- Union has members of different data types
- Hold data of only one member at a time
- Therefore, it has a memory equal to the largest member among the members of the union
- The different members share the same memory location

Union

```
union serialportData {
      char xdata projectName[];
      char  ;
      int data rate;
     };
```

Union Use in Main

- /* Main function */
- int main ( ) {
- **union** serialportData *new*; /* Declare new union having fields and data-type members equivalent to union serialportData declared earlier. The fields were empty in serialportData. */

Union Use

- new.projectName = “LED Blinking Program”; /*new.projectName = “LED BlinkingProgram”*/
- printf (“Project Name:%c \n”, new.projectName);
- new.rate = 4;

  .
- .}

Pointer

1. char *data x;   /* x is pointing to a variable value and is of *data* memory type. The data-type is char of 8-bit. When * is present before the x then x is an 8-bit address of byte as data in internal RAM in 8051. The value is pointed by that 8-bit direct address. This is because the data type is declared as data. The declaration is as per memory type declaration new method.*/

Pointer Example

2. data char *x;   /* Earlier method of memory type declaration*/

Null pointer

- The last element of a list of items, is declared as NULL

#define NULL (void*) 0x0000/* will declare the address 0x0000 as pointing to the null (nothing or no significant value)*/

Use of pointers to data or function

1. unsigned long **time_date*/* It means *time_date* is a pointer and *time_date means contents at the address identified by the time_date*/

Use of pointers to data or function

2. # define portP1 (unsigned byte *) 0x90/* It means the portP1 is assigned the address 90H. *portP1 will refer to the port P1 byte, which can be assigned any value between 0 and 255. &portP1 as an argument will pass the reference to the port P1 byte and *time_date* + 1 will refer to the next address after the 8 bytes reserved for *time_date**/

Generic Pointers

- Memory saves a generic pointer in Keil C51 compiler using three bytes
- The pointer specifications independent of the memory internal or external or specific location

Generic Pointers

- The first byte— specifies the memory type
- The second byte — the high-order byte of the offset value
- The byte 3 — the low-order byte of the offset value. [An address has two components: base address plus offset. Adding base + offset gives the physical address.]

Generic Pointers

- char *projectName; /* A pointer projectName, which points to set of addresses, which save the string of characters for the projectNameString "Serial Port Program". Pointer saves in the internal data memory of the 8051 */

Generic Pointer

- int *numTicksptr; /* A pointer numTicksptr, which points to set of addresses, which save the 16-bits for the integer variable numTicksptr. Pointer saves in the internal data memory of the 8051. */

Generic Pointer

- long *Counts; /* A pointer Counts, which points to set of 4 addresses, which save the 32-bit long integer for the counts. Pointer saves in the internal data memory of the 8051. */

Memory-Specific Pointer

- Memory-specific pointers always include a memory type specification in the pointer declaration
- The pointer then saves in a specific memory type area

Memory-Specific Pointer

1. char data *projectName; /* A pointer projectName, which points to set of addresses, which save the string of characters for the projectNameString “Serial Port Program”. Pointer saves in the internal data directly addressable memory of the 8051. This is because pointer memory type has now been explicitly specified. */

Memory-Specific Pointer

1. int xdata *numTicksptr; /* A pointer numTicksptr, which points to set of addresses, which save the 16-bits for the integer variable numTicksptr. Pointer saves in the external idata memory of the 8051. This is because pointer memory type has now been explicitly specified. */

Memory-Specific Pointer

1. long idata *Counts; /* A pointer Counts, which points to set of 4 addresses, which save the 32-bit long integer for the counts. Pointer saves in the internal idata memory (indirectly addressable memory of the 8051. This is because pointer memory type has now been explicitly specified.*/

Memory Specific Pointer

- Saves in Keil C51 compiler using one or two bytes
- Specifications dependent of the memory internal or external or specific location
- First byte—byte address in case of data or idata, bdata or pdata or lower byte of address in case of xdata or code
- Byte 2— the high-order byte of the address in case of xdata or code is memory type specified for that pointer

Function Pointers

- The address of a function stored in data memory
- Points to the start address of the function codes
- First byte of the code is at that address

Function Pointer

- functAddr = (void *) functName;

Summary

We learnt

- Array of variables
- Examples of the arrays
- Struc and Union
- Examples of Structure and union
- Pointer to point to the address of a variable, data structure or function

We learnt

- Null pointer
- Generic Pointer, like in C
- Memory specific Pointer
- Function Pointer

End of Lesson 03 on

Array, Structure, Union and Pointer